Some Utility Applications Of The Dictionary Tables in PROC SQL

Jack Hamilton

First Health
West Sacramento, California
JackHamilton@FirstHealth.com

What Are The Dictionary Tables?

- The Dictionary Tables in SQL are maintained by the SAS System.
- They contain information about various user and system objects.
- They're called tables in the documentation, but in some sense they're really views - they’re not stored anywhere.
- Dictionary Tables are available only in PROC SQL, but views of all the tables are available through the SASHELP library.

When To Use the Dictionary Tables

- Sometimes you need information about the structure of your data.
- Dictionary tables are one source of this information.
- Other sources are PROC CONTENTS, PROC CATALOG, and PROC FORMAT, and the data step information functions.

How To Specify The Dictionary Tables

A dictionary table is used just like any other table in PROC SQL:

```
select  *
from    dictionary.tables;
```

SASHELP views are used like any other views:

```
proc print data=sashelp.vtable;
run;
```

Either piece of code would print a lot of information about all available tables.

The SQL Tables and SASHELP Views

- The following pages list the names of the SQL dictionary tables, along with their SASHELP view equivalents.
- A complete list of tables can be found in the online help for PROC SQL.
- A complete list of SASHELP views can be created with the following code:

```
proc print data=sashelp.vsview;
 where libname = 'SASHELP' and
  substr(memname, 1, 1) = 'V';
run;
```

Brief Description of the Tables

- **CATALOGS** describes every member of every catalog in every library.
- **COLUMNS** describes every column in every table or view.
- **EXTFILES** describes every defined fileref, showing the path and engine.
- **INDEXES** describes every index on every dataset in every library.

Brief Description of the Tables

- **MACROS** describes every active macro variable.
- **MEMBERS** describes objects shared by tables, views, and catalogs
- **TITLES** contains the current titles and footnotes.
- **OPTIONS** contains most of the current portable system options.
- **TABLES** describes every table and view in very library.
- **VIEWS** describes every available view.

dictionary.catalogs (sashelp.vcatalg)

LIBNAME	char(8)	Library Name
MEMNAME	char(8)	Member Name
MEMTYPE	char(8)	Member Type
OBJNAME	char(8)	Object Name
OBJTYPE	char(8)	Object Type
OBJDESC	char(40)	Object Description
MODIFIED	char(8)	Date Modified
ALIAS	char(8)	Object Alias

dictionary.columns (sashelp.vcolumn)

LIBNAME	char(8)	Library Name
MEMNAME	char(8)	Member Name
MEMTYPE	char(8)	Member Type
NAME	char(8)	Column Name
TYPE	char(4)	Column Type
LENGTH	num	Column Length
NPOS	num	Column Position
VARNUM	num	Column Number in Table
LABEL	char(40)	Column Label
FORMAT	char(16)	Column Format
INFORMAT	char(16)	Column Informat
IDXUSAGE	char(9)	Column Index Type

dictionary.extfiles (sashelp.vextfl)

```
FILEREF   char(8)    Fileref
XPATH     char(80)   Path Name
XENGINE   char(8)    Engine Name
```

- It's possible to have a path name that's longer than 80 characters, so be careful with this one.

dictionary.indexes (sashelp.vindex)

LIBNAME	char(8)	Library Name
MEMNAME	char(8)	Member Name
MEMTYPE	char(8)	Member Type
NAME	char(8)	Column Name
IDXUSAGE	char(9)	Column Index Type
INDXNAME	char(8)	Index Name
INDXPOS	num	Position of Column in Concatenated Key
NOMISS	char(3)	Nomiss Option
UNIQUE	char(3)	Unique Option

dictionary.members (sashelp.vmember)

LIBNAME	**char(8)**	**Library Name**
MEMNAME	**char(8)**	**Member Name**
MEMTYPE	**char(8)**	**Member Type**
ENGINE	**char(8)**	**Engine Name**
INDEX	**char(8)**	**Indexes**
PATH	**char(80)**	**Path Name**

- This table duplicates some of the information found in other tables. The important addition is the PATH variable, but see the note on EXTFILES.

dictionary.macros (sashelp.vmacro)

SCOPE	**char(9)**	**Macro Scope**
NAME	**char(8)**	**Macro Var Name**
OFFSET	**num**	**Offset into Macro Var**
VALUE	**char(200)**	**Macro Var Value**

- If a macro variables value is longer than 200 characters, there will be multiple entries in this table. The first will have an OFFSET of 0, the second of 200, and so forth.

dictionary.titles (sashelp.vtitle)

TYPE	**char(1)**	**Title Location**
NUMBER	**num**	**Title Number**
TEXT	**char(200)**	**Title Text**

- Only the text is stored in these variables, not the font and positioning information used by SAS/GRAPH.

dictionary.options (sashelp.voption)

```
OPTNAME  char(16)  Session Option Name
SETTING  char(200) Session Option Setting
OPTDESC  char(80)  Option Description
```

- It's worth noting that this table probably contains only portable options, not platform-dependent options. That does seem to vary slightly by release.

table dictionary.tables (sashelp.vtable)

LIBNAME	char(8)	Library Name
MEMNAME	char(8)	Member Name
MEMTYPE	char(8)	Member Type
MEMLABEL	char(40)	Dataset Label
TYPEMEM	char(8)	Dataset Type
CRDATE	num	Date Created
MODATE	num	Date Modified
NOBS	num	Number of Observations
OBSLEN	num	Observation Length
NVAR	num	Number of Variables
PROTECT	char(3)	Type of Password Prot.
COMPRESS	char(8)	Compression Routine
REUSE	char(3)	Reuse Space
BUFSIZE	num	Bufsize
DELOBS	num	Number of Deleted Obs

INDXTYPE char(9) Type of Indexes

dictionary.views
(sashelp.vview)

```
LIBNAME   char(8)    Library Name
MEMNAME   char(8)    Member Name
MEMTYPE   char(8)    Member Type
ENGINE    char(8)    Engine Name
```

How Would You Use These Tables?

- Decide what you need to do.
- Figure out how you might do whatever you want to do if you were coding it manually.
- Figure out which table contains the information you need. Note some information is contained in several tables - if you wanted the names of all datasets, for example, you could use MEMBERS, TABLES, or COLUMNS.
- Use SQL or a data step to create the code.
- Run it.

Some Useful Techniques

- Use the INTO and SEPARATED BY clauses in PROC SQL to create a macro variable.
- Use CALL EXECUTE in a data step.
- Use a FILENAME pointed to a catalog entry, and %INCLUDE the results (this is preferable to writing to a temporary disk file, which is not platform-independent).

A Sample Dataset

```
1  data one;
2    key='9'; a=1; c=2; b=.;
3    output;
4    key='0'; c=12; b=1;a=0;
5    output;
6    put all;
7    /* Shows order of vars */
8  run;

KEY=0 A=0 C=12 B=1 _ERROR_=0
_N_=1
NOTE: The data set WORK.ONE has 2
observations and 4 variables.
```

Ex. 1: Put Vars Into Alpha Order

There are several techniques for reordering the variables in a SAS dataset. This first example uses PROC SQL. You might code it manually like this:

```
proc sql;
   create table two as
      select  A,B,C,KEY
      from    one;
```

Ex. 1: Put Vars Into Alpha Order

The fields you need to create this code are

- the table name (library and member)
- the column names

This information can be obtained from the DICTIONARY.COLUMNS table.

Ex. 1: Put Vars Into Alpha Order

```
proc sql noprint;
  select name
  into   :newcmd
  separated by ','
  from   dictionary.columns
  where  libname='WORK' and
         memname='ONE'
  order  by name;
create table two as
  select &NEWCMD.
  from   one;
```

NEWCMD has the value **A, B, C, KEY**

Ex. 2: Convert Char to Numeric

Converting a large number of character variables to numeric seems to be a common task. SQL provides an easy solution:

```
create table three as
   select  input(key, best.),
             a, b, c
   from    one;
```

Again, the information you need can be obtained from DICTIONARY.COLUMNS.

Ex. 2: Convert Char to Numeric

```
select
  case type
    when 'num' then name
    else 'input(' || name || ',
         best.) as ' || name
    end
into   :newcmd separated by ','
from   dictionary.columns
where  libname='WORK'
        and memname='ONE';
create table three as
  select  &NEWCMD.
  from    one;
```

Ex. 3: Rename Variables

Suppose you have a number of variables with inconvenient names, and you want to rename them all:

```
data sample;
  retain xx0y01-xx0y10 0;
  output; stop;
run;
```

In this case, suppose you want to rename variable XX0Y01 to XXY01 and so forth.

Ex. 3: Renaming Variables

One reasonable way to do the renames is with PROC DATASETS; just changing the names doesn't require rewriting the data.

```
proc datasets library=work
nolist;
  modify sample;
  rename XX0Y01=XXY01
         XX0Y01=XXY01; /* etc. */
run; quit;
```

Once more, you need the information in DICTIONARY.COLUMNS.

Ex. 3: Renaming Variables

So here's the code. Once you understand what's it's doing, it's straightforward.

```
proc sql noprint;
  select name || '=' ||
         substr(name, 1, 2) ||
         substr(name, 4, 3)
  into :renames separated by ' '
  from dictionary.columns
  where libname='WORK' and
        memname='SAMPLE' and
        name like 'XX0Y__';
quit;
```

Ex. 3: Renaming Variables

At this point, RENAMES will have the value

```
XX0Y01=XXY01 XX0Y02=XXY02 XX0Y03=XXY03 XX0Y04=XXY04
XX0Y05=XXY05 XX0Y06=XXY06 XX0Y07=XXY07 XX0Y08=XXY08
XX0Y09=XXY09 XX0Y10=XXY10
```

You can then use it in PROC DATASETS:

```
proc datasets library=work
nolist;
  modify sample;
  rename &RENAMES.;
run; quit;
```

Ex. 4: List Formats

For documentation, it might be helpful to have a list of all formats in a library. You can get that with SASHELP.VCATALG:

```
proc print data=sashelp.vcatalg;
   where objtype in
   ('FORMAT', 'FORMATC', 'INFMT',
    'INFMTC');
run;
```

This information must be combined with the list of catalogs specified in the FMTSEARCH option to find all available formats.

Ex. 5: Place Varnames Into Macvar

It's occasionally useful to have a macro variable containing the names of all variables in a dataset, except specified ones (see the FIRST./LAST. example in my paper *The Problem With NODUPLICATES, Continued*):

```
proc sql noprint;
 select  name
 into  :vnames separated by ' '
 from  dictionary.columns
 where libname = 'SASUSER'
         and memname = 'IRIS'
         and name not in
          ('SEPALLEN','PETALLEN');
```

Cautions

- The examples in this paper don't do any error checking. You should.
- Make sure your code works correctly if nothing is selected by the WHERE clause. You probably need to set the macro variable to blank before you start.
- Not all tables and views are available in all releases of SAS.
- The COLUMNS table can get very large, especially if you have SAS/GRAPH maps. Apply a WHERE clause as soon as you can.